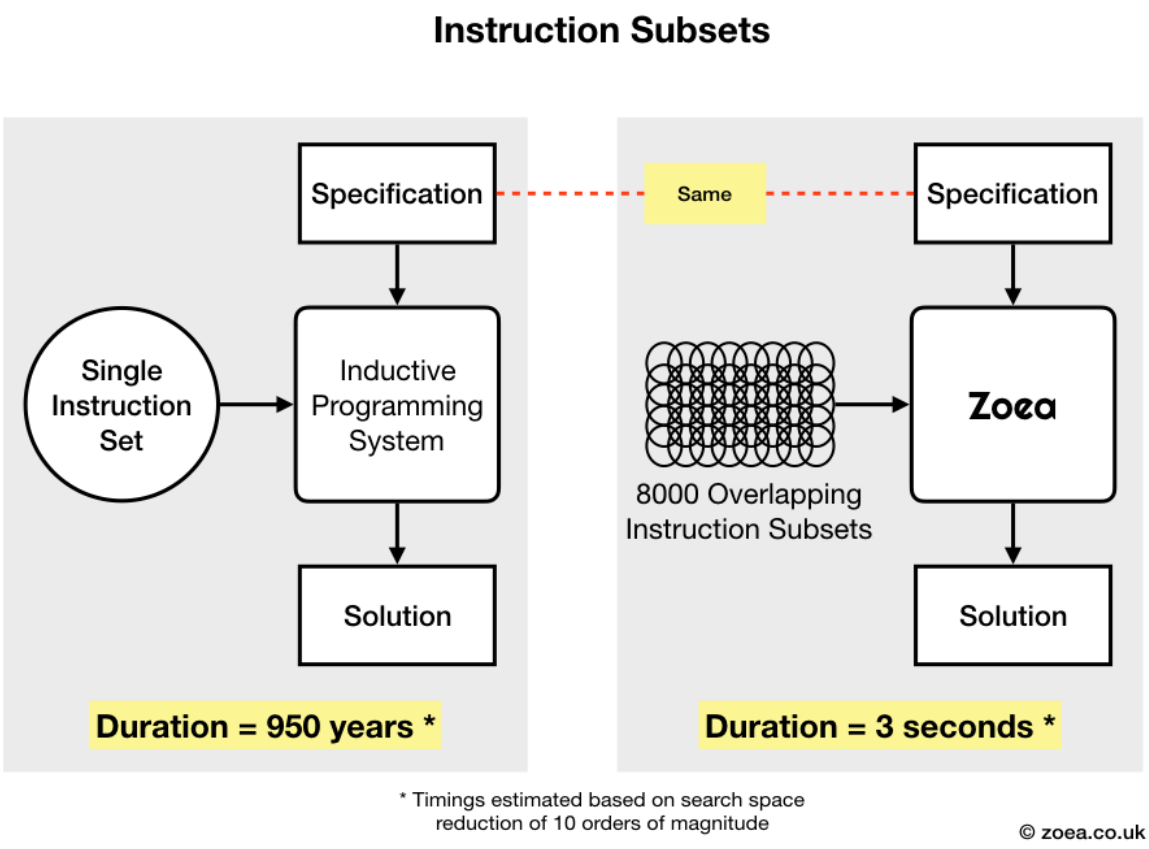
Zoea

Research note 25

Defusing the Combinatorial Explosion

Edward McDaid & Sarah McDaid
10 Mar 2023

The combinatorial explosion has been one of the biggest challenges in AI since its inception. Now, Zoea has found a way to dramatically shrink the problem.



AI uses lots of different techniques to come up with solutions. One of the most fundamental approaches in AI involves producing a tree of possible future states. For example, when playing a game like chess an AI might consider the state of the board as it is now as well as all the states corresponding to every possible move it can make. At

each of those future states the opponent could also make many different subsequent moves, and so on. This forms a tree of board states that quickly becomes impossibly large after only a few moves. The size of such trees makes it virtually impossible to find a solution. This is an example of the combinatorial explosion.

Practitioners have known about the combinatorial explosion since the earliest days of AI and various strategies have been devised to try to address it. Most of these rely on some way of recognising a 'better' or 'worse' state so we can focus on exploring only the parts of the tree that look like they might lead to a solution. In our chess example we can come up with some way of scoring each state of the board that reflects whether we are winning or losing. Then, for example, we could ignore any moves that would put us in a worse position. But this sort of approach does not work for all problems.

In a different domain, if we want to create all of the programs that are possible for a set of test case inputs we also end up with a similar tree. We do this by applying all of the programming language instructions that we can at each level to produce new values - first with the inputs and then with all the new values we produce, and so on. This tree also grows very quickly - just like the chess example. However, in this case there is no way to reliably determine if any of the states are better or worse than any other - at least not one that will work for all types of programs and data. This is the sort of problem that most AI experts wrote off as basically impossible a long time ago. However, it turns out that it isn't.

The key to making progress here was an idea we came up with just after we started working on Zoea. All programs are composed of instructions and there are around 200 instructions in most programming languages. However, the vast majority of individual programs use very few instructions - 10 or less. The idea was that if we could somehow correctly guess the instructions required for a given problem then the process of determining the program would be a lot easier. That was a great idea but there wasn't an obvious way to ensure that we could guess correctly - even with many attempts. That insight would come from somewhere else.

Zoea was designed to use many computer cores so it can work on different parts of a problem at the same time. One of the ways that Zoea breaks up a problem into pieces is to consider different subsets of the instruction set. The current instruction subsets in Zoea

mean that it can efficiently utilise up to about 100 cores. Now that we can also run Zoea on cloud based platforms we have the opportunity to use 100s or even 1000s of cores. However, this would require similar numbers of instruction subsets. That is too many to define manually so instead we decided to derive the subsets automatically from real code.

We did this using around 15.75 million lines of code. At this point it is worth emphasising that we weren't interested in the code itself - just the subsets of instructions it contained. That doesn't represent anyone's intellectual property because very many of the subsets are duplicates or subsets of one another. If instruction subsets represented intellectual property then virtually all existing code would already be illegal.

In the end we produced a number of sets of overlapping instruction subsets of different sizes. These include around 6000 subsets that contain 10 instructions and nearly 8000 subsets that contain 20 instructions. We also validated that these subsets are capable of being used to generate a very high percentage of code that wasn't used to produce them. This is exactly the result we wanted.

Using the subsets Zoea will be able to use 100s or possibly 1000s of cores to work on different parts of problems in parallel. In effect, the instruction subsets are different guesses about what instructions are required. We also know there is a very high probability that at least one of these guesses - and often many of them - will be correct. Furthermore, each core has a much easier job to do since it is only using a tiny part of the complete instruction set. This means it can produce much bigger programs in a given time. Mission accomplished. Only, this wasn't everything.

When we calculated the total size of the search trees for all the subsets and compared this with the size of the original tree for the complete instruction set we found that the total size of the search space had shrunk by between 5 and 20 orders of magnitude - depending on the size of the subsets and the size of the target program. Ten orders of magnitude is the difference between a problem taking around 950 years to solve on a computer, versus 3 seconds.

The reason for this reduction is that people don't use all combinations of instructions with equal probability but rather the distribution is highly skewed. Most of the search tree for

programs corresponds to code that people would never write and that is effectively excluded by the subsets. Also, the size of the subsets limits the branching factor from around 200 to between 10 and 50. In any event, reducing the size of the search tree equates to corresponding reductions in the amount of time and effort that is required.

A paper describing this study and including key results has been peer reviewed and was recently presented at the ICAART 2023 AI conference in Lisbon. We also determined that the proportion of duplicated effort due to subsets overlapping quickly becomes insignificant as program size increases.

This approach represents a massive reduction in the combinatorial explosion with regard to inductive programming. It may not be a completely generic solution to the problem but it - or something very similar - should be applicable in other areas of AI and computer science. Also, unlike many advances in AI, the approach is so simple it can quickly be understood by almost anyone. Once grasped, it also seems obvious. In a way it has been hiding in plain sight the whole time.

Beyond a game-changing breakthrough in inductive programming this whole story seems to be telling us two things. Firstly, traditional approaches to AI have much more mileage left in them than their current level of popularity would seem to imply. Also, the combinatorial explosion is not as scary and intractable as everybody always thought.

Learn more at **zoea.co.uk**